# LaMUX: Optimized Logic-Gate-Enabled High-Performance Microfluidic Multiplexer Design

Siyuan Liang[1], Yushen Zhang[2], Rana Altay[3], Hudson Gasvoda[3], Mengchu Li[2], Ismail Emre Araci[3], Tsun-Ming Tseng[2], Ulf Schlichtmann[2], Tsung-Yi Ho[1]

[1] The Chinese University of Hong Kong, [2] Technical University of Munich, [3] Santa Clara University

## ABSTRACT

After decades of development, flow-based microfluidic biochips have become an increasingly attractive platform for biochemical experiments. The fluid transportation and the on-chip device operation are controlled by microvalves, which are driven by external pneumatic controllers. To meet the increasingly complex experimental demands, the number of microvalves has significantly increased, making it necessary to adopt multiplexers (MUXes) for the actuation of microvalves. However, existing MUX designs have limited coding capacities, resulting in area overhead and excessive chip-to-world interface. This paper proposes a novel gate structure for modifying the current MUX architecture, along with a mixed coding strategy that achieves the maximum coding capacity within the modified MUX architecture. Additionally, an efficient synthesis tool for the mixed-coding-based MUXes (LaMUXes) is presented. Experimental results demonstrate that the LaMUX is exceptionally efficient, substantially reducing the usage of pneumatic controllers and microvalves compared to existing MUX designs.

## 1 INTRODUCTION

Flow-based microfluidic biochips have become one of the most promising platforms for biochemical experiments [1]. Such coin-sized chips can incorporate many miniaturized devices to carry out complex operations, which are traditionally performed in cumbersome laboratory instruments [2]. Thus, these chips enjoy many advantages including small consumption of reagents, increased automation degree, and reduced manufacturing costs [3]. Such a chip consists of two layers: a flow layer containing the flow channels and a control layer containing the control channels. The fluid transportation in the flow channels and the operation of on-chip devices are controlled by microvalves, which are tiny switches built at the intersections of the flow channels and the control channels [4]. Figure 1(a) shows the 3D schematic of a microvalve and the corresponding channel connections. And as shown in Figure 1(b), we can transport pressure to the control channel to control the microvalve. When the pressure is low, the microvalve is open, and fluids in the flow channel can safely pass it. By contrast, when the pressure is high, the inflated control channel will squeeze the flow channel, making the microvalve closed and block the fluid movement.



*DAC '24, June 23–27, 2024, San Francisco, CA, USA*



ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3655942

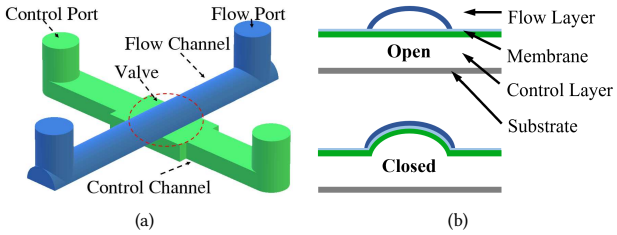


**Figure 1: Schematics of a microvalve. (a) The three-dimensional schematic. (b) The cross-section schematic.**

One of the main challenges in the development of microfluidic biochips is to reduce the dependency on external pneumatic controllers. Specifically, the fluid and pressure are transported between on-chip and off-chip components via ports, which are punch holes on the chip. Each port consumes remarkable chip area and requires an external pneumatic controller. As the complexity of experimental demands increases, the number of microvalves increases significantly, making it impractical to assign an independent port to every microvalve. Hence, it has become necessary to adopt multiplexers (MUXes) to control the microvalves [4].

The classic MUX design was proposed by Thorsen et al. [2], which can address $2^N$ control channels individually with $2N$ independent pneumatic controllers and ports based on binary coding. Based on the classic MUX design, a powerful improvement was proposed by Liang et al. [4], which can use $2N$ independent pneumatic controllers and ports to address up to $C_N^{2N}$ control channels based on combinatorial coding. However, current MUX designs have limited coding capacities, and require more pneumatic controllers and flow ports than necessary, which leads to area overhead and excessive chip-to-world interface. For example, in middle-scale designs, the MUX can easily consume nearly half of the chip area [5, 6]. Hence, new MUX designs that can address more control channels using the same number of pneumatic controllers and flow ports are in great demand.

In current MUX designs, each flow channel is assigned with an independent pneumatic controller and flow port, and has two states: *depressurized* and *pressurized*, which can be utilized to address control channels. In this paper, we propose a novel triple state gate named $T$ gate, which enables each pneumatic controller to provide three states: (*depressurized*, *pressurized*), (*pressurized*, *depressurized*), and (*pressurized*, *pressurized*). By applying the gate to the state-of-the-art MUX architecture, we propose a new MUX architecture, in which every two adjacent flow channels are connected to the flow input and output of a $T$ gate, and are controlled by a single pneumatic controller. Hence, compared to the state-of-the-art MUX architecture with the same number of flow channels,

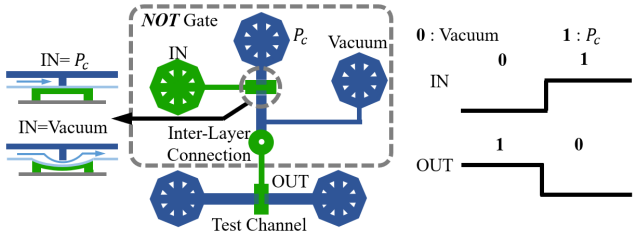


**Figure 2: The schematic of the classic microfluidic *NOT* gate [8], and the output variations depending on the input.**

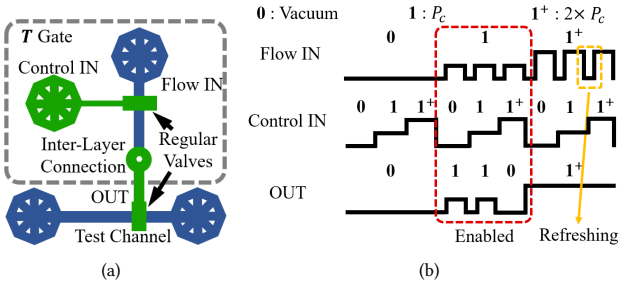


**Figure 3: The proposed gate. (a) The schematic of the proposed $T$ gate. (b) Output variations depending on the inputs.**

our MUX architecture only requires half the number of pneumatic controllers and flow ports. We then propose a mixed coding strategy, and a mixed-coding strategy based MUX design (LaMUX). The LaMUX achieves the provably maximum coding capacity within our MUX structure. Moreover, we propose an algorithm to efficiently synthesize the LaMUX of arbitrary sizes according to user demands, and minimize the microvalve usage. According to experimental results, the mixed coding strategy can provide much more codes than existing coding strategies, thus significantly reducing the usage of pneumatic controllers in LaMUXes.

## 2 MICROFLUIDIC LOGIC GATE

Logic gates, which can be used to generate additional signals with different patterns than the input, and thus reducing the dependence on pneumatic controllers, are important components in microfluidic biochips. Different from the electronic circuit, which can directly use the electric potential of single wire to denote 0 and 1, flow-based microfluidic circuits need to rely on the pressure difference between channels to represent 0 and 1 [7]. In this section, we will first introduce the classic *NOT* gate design, and then introduce our proposed $T$ gate.

### 2.1 The classic *NOT* gate

Figure 2 shows the schematic of a microfluidic *NOT* gate design [8]. This design uses 1 control port as the input, and 2 flow ports connected to constant pressure sources of the control pressure $P_c$ and vacuum, respectively. Before the output, there is an inter-layer connection to transport the pressure in the flow channel to the follow-up control channel, thus making it possible to utilize the output signal to conduct further control. At the intersection of the input control channel and the flow channel, the designers add a bump in the flow channel, which is high enough to reach the top of the control channel. In this case, when the input is $P_c$, the pressure difference on both sides of the membrane at the intersection is 0, so the control channel will not deform and the bump will seal the flow channel, making the output vacuum; on the contrary, when the input is vacuum, the pressure difference at the intersection will make the control channel shrink and open the flow channel, making the output $P_c$.

There are two drawbacks that hinder the classic *NOT* gate's application. Firstly, the bump structure added at the intersection of the control channel and the flow channel requires special treatment during fabrication to alter the shape of the flow channel. Secondly, since the two flow ports are connected to constant pressure sources of $P_c$ and vacuum, denoted as 1 and 0, the shape of the flow channels requires careful design for an appropriate fluid impedance distribution [8], otherwise the pressure might directly get transported to the vacuum flow port instead of the output under all conditions.

### 2.2 The proposed $T$ gate design

A fact that has been neglected by many previous designers is that the pneumatic controllers are actually capable of providing multiple levels of pressure [9]. We let the pneumatic controllers provide two levels of pressure: $P_c$ and $2 \times P_c$, denoted as 1 and $1^+$, and propose a novel gate design, which is shown in Figure 3(a). The proposed gate design only uses regular microvalves, reduces flow port usage, and does not require careful design of fluid impedance, which makes it easier to apply this gate in real chips.

Figure 3(b) shows the variations of the output when adding different pressures at the flow input and the control input. Different from the classic gate, when switching to a new state, we need to first set the flow input briefly to 0 for refreshing, and then input the new pressure for it after the control input is updated. When the flow input is $P_c$, the output will vary according to the control input, we call the gate is enabled at this time; otherwise, the gate is not enabled, and the output will keep the same as the flow input regardless of the control input. Focusing on the case that the gate is enabled, there are three states: When the control input is 1, there is no pressure difference on both sides of the microvalve, so that the output will follow the input to be 1. Otherwise, when the control input is 0, the microvalve will be open, and the output will become 1; when the control input is $1^+$, the microvalves will be closed, and the output will become 0, which is the same performance of a *NOT* gate. Since the proposed gate has one more state than the classic *NOT* gate, we name it the triple state gate ($T$ gate).

## 3 MICROFLUIDIC MULTIPLEXER

### 3.1 The state-of-the-art multiplexer

Different from the flow circuits for biochemical experiments, the MUX uses flow channels to address control channels. Figure 4 shows the schematic of an example CoMUX, which is the state-of-the-art microfluidic MUX design.

The flow channels and control channels are denoted by blue and green lines, respectively. Microvalves are denoted by green rectangles, those with black crosses are being closed and others are open. In this architecture, flow ports are arranged in two rows on the left and right sides, while maintaining a distance of 2 *mm*

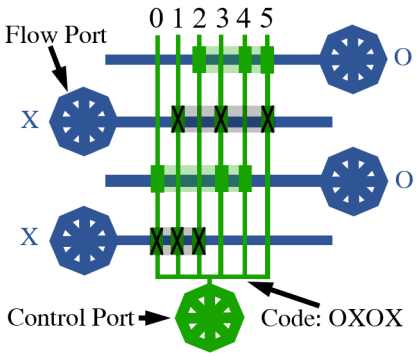


**Figure 4: The schematic of a CoMUX that uses 4 pneumatic controllers to individually address 6 control channels.**

between ports in the same row according to the design rule [10]. The multiplexing mechanism of the CoMUX can be summarized as:

- Each flow channel is controlled by an independent pneumatic controller, and has two states. It can be depressurized or pressurized, denoted as $O$ and $X$, respectively. When a flow channel is depressurized, all microvalves along it will be open; otherwise, all microvalves along it will be closed.
- Each control channel has a unique code, which is a sequence of states $O$, $X$. And for a CoMUX with $N$ flow channels, each code must have $\lfloor \frac{N}{2} \rfloor$ $O$s.
- The value of the $i^{th}$ element of the control channel code determines the exact position of the microvalve. If it is $O$, the microvalve will be placed at the intersection with the $i^{th}$ flow channel; otherwise, no microvalves will be placed at the corresponding intersection.
- As shown in Figure 4, when inputting the control sequence of $OXOX$ into the flow ports, at least one microvalve will be closed in all control channels except for control channel 4, whose corresponding code is consistent with the control sequence.

In general, a CoMUX collects all codes with $\lfloor \frac{N}{2} \rfloor$ $O$s, which is called the combinatorial coding, and uses an independent pneumatic controller to control each flow channel. Hence, the CoMUX can use $N$ pneumatic controllers to address $C^N_{\lfloor \frac{N}{2} \rfloor}$ control channels. An efficient channel merging approach was also proposed in the CoMUX paper to reduce the microvalve usage in the MUX [4].

## 3.2 The proposed multiplexer architecture

We make use of the dual $T$ gate structure, whose flow layer components and control layer components are reversed from the original $T$ gate, to modify the state-of-the-art MUX architecture and enhance its ability to address more control channels.

Figure 5 shows our MUX architecture, in which each *Flow IN* is connected to a dual $T$ gate, another flow channel is connected to the output of the gate and placed parallel to the previous flow channel. The dual $T$ gates on the two sides of the MUX share two control ports, respectively, which are then connected to the same constant pressure source with $P_c$ to enable the $T$ gates, so that the two control ports can be regarded as the same *Control IN*. The to-be-addressed control channels are supplied with $P_m$, which is the control pressure in the main functional part of the design. Here

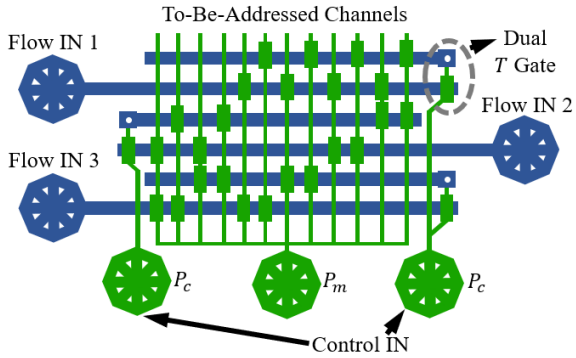


**Figure 5: The schematic of the proposed MUX architecture with 3 flow ports.**

$P_c \geq P_m + \triangle P_v$, where $\triangle P_v$ is the minimum pressure difference to close a microvalve. It is worth mentioning that when briefly turning off the pressure source of $P_c$ for refreshing, the pressure source of $P_m$ will also be needed to get briefly turned off to prevent control mistakes in the main functional part of the design. Meanwhile, a pressure regulator can be used here to obtain $P_m$ from the pressure source of $P_c$ [11], so that the three control ports can be actually connected to the same pressure source if needed.

The state-of-the-art MUX architecture assigns two independent pneumatic controllers for two adjacent flow channels, and can provide four states: $OX$, $XO$, $XX$, and $OO$. In our MUX architecture, with the help of the dual $T$ gate, we just need one pneumatic controller for the control of two adjacent flow channels, and can have three states $OX$, $XO$, and $XX$ by inputting different pressures into the *Flow IN*. Hence, our MUX architecture can make better use of the pneumatic controllers, and has stronger ability to address more control channels.

# 4 MIXED CODING STRATEGY

## 4.1 Coding problem formulation

In our MUX architecture, we continue to use the setting in the current MUX design, and define a *code* of a control channel as a sequence of binary values $O, X$. For a specific control channel in the MUX, if the $i^{th}$ element of its code is $O$, there will be a microvalve at the intersection between the control channel and the $i^{th}$ flow channel; otherwise, there will be no microvalve at the corresponding intersection.

Facing our architecture, it is natural to wonder: *What is the maximum coding capacity of this architecture, and what coding strategy we should use to achieve that?* To answer these questions, we need to first specify the design rules of the coding strategy:

(1) Every control channel requires a unique code.
(2) When triggering the code of a control channel, all microvalves along it must be open, while at least one microvalve along every other control channel must be closed.

Here, by *triggering a code*, we refer to the operation that we pressurize/depressurize the flow channels in the MUX according to the code, i.e., flow channels marked by $O$ are depressurized, and flow channels marked by $X$ are pressurized.

Since we make use of the dual $T$ gate to enable each pneumatic controller to control a flow channel pair in our MUX, to figure

out the maximum number of control channels that $N$ independent pneumatic controllers can address, we need to find the maximum number of feasible codes of length $2 \times N$. Subsequently, we examine the problem through the lens of set theory, thereby enhancing the lucidity of the matter at hand. For each code $c$, we construct:

- a set $\mathcal{T}_c$ containing the indices of the flow channels that are marked by $O$;
- a set $\overline{\mathcal{T}_c}$ containing the indices of the flow channels that are marked by $X$.

Then the design rules can be formulated as:

$$\begin{aligned} &\forall \mathcal{T}_c, \mathcal{T}_{c'} \in \{1, 2, \cdots, 2N\} \; and \; c \neq c' \\ &\mathcal{T}_c \neq \mathcal{T}_{c'}, and \; \mathcal{T}_c \cap \overline{\mathcal{T}_{c'}} \neq \emptyset \end{aligned} \tag{1}$$

Specifically, the second design rule can be understood as triggering code $c$ will not trigger other codes $c'$ at the same time, so the constraint can be further formulated as:

$$\forall \mathcal{T}_c, \mathcal{T}_{c'} \in \{1, 2, \cdots, 2N\}, and \; c \neq c', \mathcal{T}_c \nsubseteq \mathcal{T}_{c'} \tag{2}$$

Additionally, considering the working scheme of the dual $T$ gate, the pressures of each flow channel pair can only be $OX$, $XO$ or $XX$. So here exists an extra constraint:

$$\begin{aligned} &\forall k \in \{1, 2, \cdots, N\}, and \; \forall \mathcal{T}_c \in \{1, 2, \cdots, 2N\} \\ &if \; 2k-1 \in \mathcal{T}_c : 2k \notin \mathcal{T}_c; \; if \; 2k \in \mathcal{T}_c : 2k-1 \notin \mathcal{T}_c. \end{aligned} \tag{3}$$

## 4.2 Mixed coding strategy

For any code $c$, if we let the pressures of other flow channel pairs keep the same, and only change the pressures of one flow channel pair from $OX$ to $XO$ to derive a new code $c'$, the corresponding $\mathcal{T}_c$ and $\mathcal{T}_{c'}$ will not be subset of each other, and can be assigned to control channels in the same MUX; But if we change the pressures of the channel pair from $XX$ to either $OX$ or $XO$ to derive a new code $c''$, $\mathcal{T}_c$ will become a subset of $\mathcal{T}_{c''}$, so that code $c$ and $c''$ can not be assigned to control channels in the same MUX.

Hence, if we only use codes whose $|\mathcal{T}_c| = a$ in our MUX, we can have a coding capacity of at most $C_a^N \times 2^a$, and we use $\mathcal{T}_{c_i}^a$ to denote every such code. Now move on to a more complex case of using codes of two different lengths in the same MUX, assuming we wish to add other codes whose $|\mathcal{T}_c| = b$, $b \neq a$ without violating design rules, we need to guarantee:

$$\begin{aligned} S_a &= \mathcal{T}_{c_1}^a \cup \mathcal{T}_{c_2}^a \cup \cdots \cup \mathcal{T}_{c_i}^a \\ S_b &= \mathcal{T}_{c_1}^b \cup \mathcal{T}_{c_2}^b \cup \cdots \cup \mathcal{T}_{c_i}^b \\ S_a \cap S_b = \emptyset, &S_a \cup S_b = \{1, 2, \cdots, 2N\} \end{aligned} \tag{4}$$

In other words, we form $\mathcal{T}_{c_i}^a$ and $\mathcal{T}_{c_i}^b$ from non-intersecting subsets of the $N$ flow channel pairs, respectively. In this case, if we let $|S_a| = x$, then the maximum coding capacity can be achieved will become $C_a^x \times 2^a + C_b^{N-x} \times 2^b$. Then we can derive the most general case of using codes of all lengths, and the problem of finding the maximum coding capacity becomes:

$$\begin{aligned} \max \; & C_1^{x_1} \times 2^1 + C_2^{x_2} \times 2^2 + \cdots + C_N^{x_N} \times 2^N \\ s.t. \quad & x_1 + x_2 + \cdots + x_N = N \\ & x_i \in \{0\} \cup \{i, i+1, \cdots, N\} \end{aligned} \tag{5}$$

**Table 1: Solutions of the Dynamic Programming**

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $i_{argmax}$ | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 |
| Coding Capacity | 2 | 4 | 12 | 32 | 80 | 240 | 672 | 1792 | 5376 | 15360 |

This is a discrete optimization problem, which can be efficiently solved by dynamic programming [12]. We set up the corresponding dynamic programming as:

$$\begin{aligned} dp(i, N) &= \max_{x_i} \left( dp(i-1, N-x_i) + cost(x_i) \right) \\ cost(x_i) &= \begin{cases} 0 & if \; x_i < i; \\ C_i^{x_i} \times 2^i & otherwise. \end{cases} \end{aligned} \tag{6}$$

After solving the problem for different values of $N$, we notice that the maximum coding capacity occurs when only one $x_i = N$. Table 1 shows the solutions of the dynamic programming, the second row shows the corresponding $i$ of the $x_i = N$ that can lead to the maximum coding capacity, and the third row shows the corresponding maximum coding capacity. Hence, for $N$ pneumatic controllers, we decide to enumerate all $\mathcal{T}_c^{i_{argmax}}$, and form codes accordingly. In this process, we not only need to find all combinations of $i_{argmax}$ flow channel pairs, which is a combinatorial operation, but also need to do binary traversal in each combination. Hence, we name it the *mixed coding strategy*.

# 5 SYNTHESIS TOOL FOR LAMUX

To be distinguished from existing MUX designs, we refer to a MUX applying the mixed coding strategy as a LaMUX. Since the mixed coding strategy does not specify the order of the control channels, we can design different LaMUXes by exchanging the order of control channels. Hence, in this section, we propose a synthesis tool for LaMUX, so that the usage of microvalves and total channel length in the LaMUX is minimized.

## 5.1 Selection of suitable $N$ and $|\mathcal{T}_c|$

Given the number of to-be-addressed control channels, denoted as $N_C$, we need to first determine the number of flow channel pairs, denoted as $N$, and the number of microvalves per control channel, denoted as $|\mathcal{T}_c|$. Following the proposed mixed coding strategy, we choose the smallest possible $N$ with $2^{i_{argmax}} \times C_{i_{argmax}}^N \geq N_C$, and let $|\mathcal{T}_c| = i_{argmax}$.

## 5.2 Code generation

After determining the value of $N$ and $|\mathcal{T}_c|$, we develop a 2-stage algorithm to generate all the codes provided by the proposed mixed coding strategy.

*5.2.1 Combinatorial traverse.* Firstly, we need to find all combinations of $i_{argmax}$ flow channel pairs among these $N$ flow channel pairs, which is a combinatorial traverse process.

We build up a list $MEM$ of size $i_{argmax}$ to enumerate the combinations, and we use $v(i)$ to indicate the value of the $i^{th}$ element. We initialize $MEM$ with $[1, 2, \cdots, i_{argmax}]$, i.e., $v(j) = j$, as the first combination. Then, we will iteratively produce the next combination based on the current combination stored in $MEM$, until all the $C_{i_{argmax}}^N$ possible combinations are generated.

There are two functions used in the combinatorial traverse algorithm: *INCREASE* and *RESET*. For *INCREASE*, it attempts to

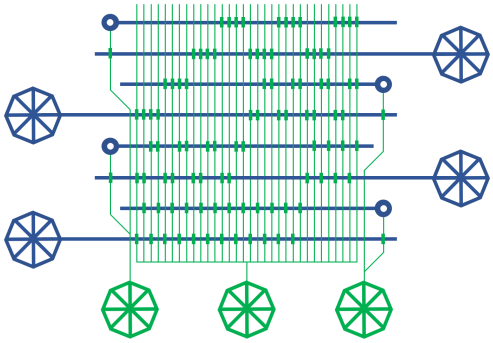

**Figure 6: An initial LaMUX addressing 32 control channels.**

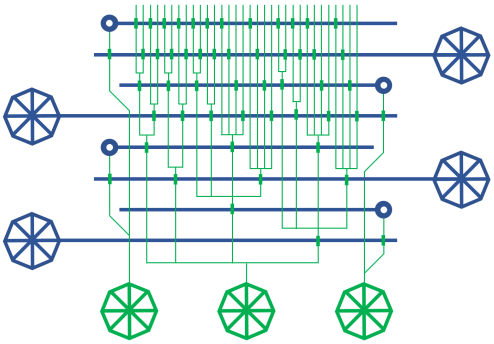

**Figure 7: The corresponding optimized LaMUX of the design shown in Figure 6.**

increase $v(i_{argmax})$ by one. For *RESET*, it examines $v(j)$ sequentially in a reverse order from $j = i_{argmax}$ to $j = 1$. If there is any specific $v(q)$ such that $v(q)+1 < v(q+1)$, all the $v(j)$ from $j = q$ to $j = i_{argmax}$ will be set to $v(q)+1, v(q)+2, \cdots, v(q)+i_{argmax}-q+1$.

The proposed algorithm performs *INCREASE* and *RESET* iteratively. In each iteration, we repeatedly perform *INCREASE* until $v(i_{argmax}) = N$, and then perform *RESET*. The entire algorithm terminates when *RESET* fails to find any $v(q)$ that fulfills $v(q) + 1 < v(q + 1)$. Every time *INCREASE* or *RESET* is successfully executed, we obtain a valid combination of flow channel pairs, which can be transformed to a set of codes in the next stage.

*5.2.2 Binary traverse.* As illustrated in section 3.4, for an arbitrary combination of flow channel pairs, by changing the pressures of any flow channel pair in the combination from $XO$ to $OX$, we can derive a new code that has no conflict with the current code. Hence, for each combination of $i_{argmax}$ flow channel pairs, we can generate $2^{i_{argmax}}$ codes that have no conflict with each other.

We regard $XO$ as 1, and $OX$ as 0. For each combination of flow channels generated before, we make use of a $i_{argmax}$-bit binary number to conduct the binary traverse and derive all the $2^{i_{argmax}}$ codes corresponding to this combination. The $i_{argmax}$-bit binary number is initialized as all zeros, which leads to the first code, and we always add 1 to the $i_{argmax}$-bit binary number to derive new codes until the $i_{argmax}$-bit binary number become all ones. For each value of the $i_{argmax}$-bit binary number, we assign the corresponding pressures to the flow channel pairs within the combination, and let flow channel pairs outside the combination be $XX$ to derive a standard code. For example, let $N = 4$, $i_{argmax} = 3$, and a specific combination $\{1, 3, 4\}$. Assume the 3-bit binary number is 011 at some point in the process, then the code generated at this time is $(OX)(XX)(XO)(XO)$.

After generating all the $2^{i_{argmax}} \times C^N_{i_{argmax}}$ codes, we pick the first $N_C$ of them, and simply form control channels and microvalves accordingly to derive the initial LaMUX design. Figure 6 is an example initial LaMUX using 4 pneumatic controllers to address 32 control channels.

## 5.3 Channel merging

After deriving the initial LaMUX, we need to determine which group of channels and how long the channel segments should merge at this stage to minimize the microvalve usage in the MUX. We construct a list of size $N_C$ to record the group information of each control channel. Initially, all control channels are set to belonging to the same channel group.

Then the merging process starts from the bottom and progresses row by row, specifically per flow channel, until reaching the top. Microvalves can merge together if they meet three conditions: (1) they are located on the same flow channel, (2) they are on adjacent control channels, and (3) they still belong to the same channel group. If a control channel forms a microvalve with a specific flow channel, while another control channel does not form a microvalve with it, these two channels will be assigned to separate channel groups. It is important to note that if two adjacent microvalves already belong to different groups, they cannot merge. Following this process, we can obtain an optimized LaMUX design with significantly fewer microvalves. The example shown in Figure 7 is the corresponding optimized LaMUX derived based on Figure 6, which has significantly reduced number of microvalves.

# 6 EXPERIMENTAL RESULTS

To demonstrate the performance of the proposed method, we compare LaMUX to three existing MUX designs with two groups of experiments.

## 6.1 Performance in coding capacity

We first test the maximum number of codes that the proposed mixed coding strategy can provide, i.e., its coding capacity. We compare the mixed coding strategy with three existing coding strategies: the binary coding [2], the combinatorial coding [4], and the multi-level combinatorial coding [9] at different scales.

As shown in Figure 8, the mixed coding strategy has the largest coding capacity at all scales. Compared to the binary coding and the combinatorial coding, which only let the pneumatic controllers to provide pressures of a single level, the advantage of the mixed coding keeps growing as the number of pneumatic controllers increases. The multi-level combinatorial coding is similar to the mixed coding, because they both let the pneumatic controllers to provide 2 levels of pressures. Although the MUX based on the multi-level combinatorial coding requires careful fabrication of two different types of microvalves [9], and it is actually hard to apply this kind of MUX in real chips, we still add it into the comparison just to achieve a more comprehensive experiment. Compared to the multi-level combinatorial coding, although the lead is not as large as the other two strategies, the mixed coding can still keep providing more codes and keep the lead at all scales.

**Table 2: Comparison of Resource Usage in Different MUXes under different control demands**

| Mux Type | Performance Metrics | Number of to-Be-Addressed Control Channels ($N_C$) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10 | 20 | 40 | 80 | 160 | 320 | 640 | 1280 | 2560 | 5120 | 10240 | 20480 | 40960 | 81920 |
| Classic MUX[2] | Number of Pneumatic Controllers ($N$) | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 |
| | Number of Microvalves | 40 | 100 | 240 | 560 | 1280 | 2880 | 14080 | 30720 | 30720 | 66560 | 143360 | 307200 | 655360 | 1392640 |
| CoMUX[4] | Number of Pneumatic Controllers ($N$) | 5 | 6 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| | Number of Microvalves | 14 | 34 | 55 | 110 | 234 | 459 | 982 | 1986 | 4142 | 8806 | 17132 | 36392 | 71011 | 150489 |
| Multi-Level MUX [9] | Number of Pneumatic Controllers ($N$) | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 9 | 9 | 10 | 11 | 11 | 12 | 13 |
| | Number of Microvalves | 24 | 50 | 110 | 300 | 605 | 1320 | 3140 | 6274 | 13848 | 32186 | 63826 | 137044 | 309200 | 615892 |
| LaMUX | Number of Pneumatic Controllers ($N$) | 3 | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 9 | 9 | 10 | 11 | 11 | 12 |
| | Number of Microvalves initial | 20 | 60 | 120 | 240 | 640 | 1600 | 3200 | 6400 | 15630 | 30720 | 71680 | 143360 | 286720 | 655360 |
| | Number of Microvalves optimized | 14 | 31 | 60 | 110 | 239 | 520 | 982 | 1871 | 3993 | 7610 | 15807 | 30907 | 59520 | 123359 |

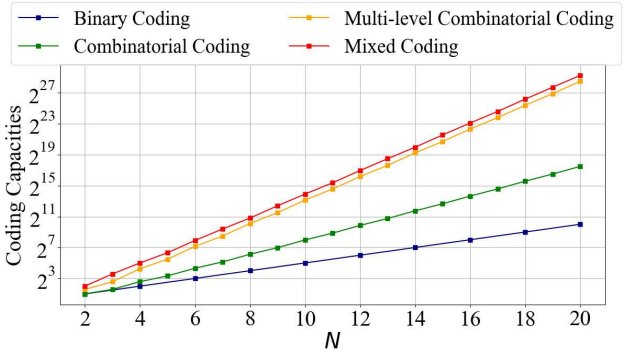


**Figure 8: Comparison of coding performance with existing methods.**

Here is an example that shows the performance of different strategies more specifically. A MUX consisting of 20 pneumatic controllers can address $1 \times 10^3$, $1.8 \times 10^5$, and $3.8 \times 10^8$ control channels by applying the three existing coding strategies. While a MUX consisting the same number of pneumatic controllers can apply the mixed coding strategy to address up to $6.4 \times 10^8$ control channels, which significantly improves the coding capability.

## 6.2 Performance in resource usage

We then evaluate the resource usage of LaMUX when addressing different number of control channels. We compare the LaMUX with three existing MUX designs: the classic MUX which uses the binary coding [2], the CoMUX which uses the combinatorial coding [4], and the multilevel MUX which uses the multi-level combinatorial coding [9]. We consider two metrics: the number of pneumatic controllers $N$ and the number of microvalves in MUXes.

The experiments starts from $N_C = 10 \times 2^0$, and we double $N_C$ for each comparison until it reaches $10 \times 2^{13}$, which is of significant size to meet the control requirements of microfluidic applications currently in use [4]. The results are shown in Table 2.

For the usage of pneumatic controllers, thanks to the great performance of the mixed coding strategy, LaMUX can save up to 65%, 35%, and 10% pneumatic controllers compared to the classic MUX, the CoMUX, and the multi-level MUX, respectively.

As for the microvalve usage in MUXes, the microvalve usage of the initial LaMUX is already better than the classic MUX, but is close to the multi-level MUX, and worse than CoMUX. By applying channel merging, the microvalve usage of the optimized LaMUX is significantly reduced, and becomes better than all of the three existing MUX designs. Specifically, the microvalve usage of LaMUX is 90%, 20%, and 80% fewer than the classic MUX, the CoMUX, and the multi-level MUX, respectively.

# 7 CONCLUSION

In this paper, we have proposed a novel MUX design, named LaMUX to address the microvalve control problem. We have proposed a novel $T$ gate to modify the state-of-the-art MUX architecture, and a mixed coding strategy to achieve the maximum coding capacity within our MUX architecture. Moreover, a synthesis tool has also been presented to generate LaMUXes of any scales with minimized number of microvalves. Experimental results have demonstrated that the LaMUX is exceptionally efficient, substantially reducing the usage of pneumatic controllers and microvalves compared to existing MUX designs.

# 8 ACKNOWLEDGMENT

The research work described in this paper was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust. This work is jointly supported by the Research Grants Council of Hong Kong SAR (No. CUHK14217022), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, No. 456006534), and the NSF program 19-582 (No. 2045087). Prototype of this device is being fabricated and tested in the Center for Nanostructures at Santa Clara University.

# REFERENCES

[1] J. M. Perkel, "Life science technologies: Microfluidics—bringing new things to life science," *Science*, vol. 322, no. 5903, pp. 975–977, 2008.
[2] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.
[3] G. M. Whitesides, "The origins and the future of microfluidics," *nature*, vol. 442, no. 7101, pp. 368–373, 2006.
[4] S. Liang, M. Li, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "Comux: Combinatorial-coding-based high-performance microfluidic control multiplexer design," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.
[5] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2017.
[6] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann, "Columba s: A scalable co-layout design automation tool for microfluidic large-scale integration," in *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.
[7] W. H. Grover, A. M. Skelley, C. N. Liu, E. T. Lagally, and R. A. Mathies, "Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices," *Sensors and Actuators B: Chemical*, vol. 89, no. 3, pp. 315–323, 2003.
[8] M. Rhee and M. A. Burns, "Microfluidic pneumatic logic circuits and digital pneumatic microprocessors for integrated microfluidic systems," *Lab on a Chip*, vol. 9, no. 21, pp. 3131–3143, 2009.
[9] D. W. Lee, I. Doh, Y. Kim, and Y.-H. Cho, "Advanced combinational microfluidic multiplexer using multiple levels of control pressures," *Lab on a chip*, vol. 13, no. 18, pp. 3658–3662, 2013.
[10] "Microfluidics design rules." https://www.stanfordmicrofluidics.com/design-basics.
[11] K. Svensson, S. Södergren, and K. Hjort, "Thermally controlled microfluidic back pressure regulator," *Scientific Reports*, vol. 12, no. 1, p. 569, 2022.
[12] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.